Attorney Docket No.: 26530.94 (IDR-685)
Customer No. 27683

EXPRESS MAIL NO. :EV333436785US DATE OF DEPOSIT: February 11, 2004

This paper and fee are being deposited with the U.S. Postal Service Express Mail Post Office to Addressee service under 37 CFR §1.10 on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

Gayle Conner
Name of person mailing paper and fee

Gayle Conner
Signature of person mailing paper and fee

METHOD AND SYSTEM FOR AUTOMATICALLY CREATING AND UPDATING ACCESS CONTROLS LISTS

Inventors: Ashutosh Vyas
s/o Mr. O.P. Vyas
B-5, UIT Colony
Pratap Nagar, Jodhpur-342004
India
Citizenship: India

Gaurav Jain
College Book House
Church Road, Alwar-301001
India
Citizenship: India

Madhusudhana H.S.
No. 48, Royal Shelters
Begur Road, Devarachikkanahalli
Bangalore, 560 076
India
Citizenship: India

Assignee: Novell, Inc.
1800 South Novell Place
M/S PRV-F-331
Provo, Utah 84606

HAYNES AND BOONE, LLP
901 Main Street, Suite 3100
Dallas, Texas 75202-3789
(214) 651-5000

Attorney Docket No. 26530.94 (IDR-685)
R: 60601.1

EXPRESS MAIL NO.: EV333436785US DATE OF DEPOSIT: February 11, 2004

This paper and fee are being deposited with the U.S. Postal Service Express Mail Post Office to Addressee service under 37 CFR §1.10 on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

Gayle Conner
Name of person mailing paper and fee

Signature of person mailing paper and fee

METHOD AND SYSTEM FOR AUTOMATICALLY CREATING AND UPDATING ACCESS CONTROLS LISTS

BACKGROUND

[0001] This disclosure relates generally to networked computing environments and, more specifically, to a method and system for automatically updating and creating access control lists.

[0002] Many computing resources now exist on networks. Files, programs, webpages, or data, for example, may be stored on a network and accessed remotely. The Internet, local area networks (LANs), wide area networks (WANs), wireless networks, and intranets, for example, may have items for which remote access is desired.

[0003] Policies dictating access rights may be used with some network resources. Access control rules may be used to enforce policies and permissions regarding access to various network resources. Such access control rules may be grouped into access control lists (ACLs). ACLs may need to be properly ordered and maintained to ensure that the ACL enforces the desired policy for the network. Generally, maintenance of ACLs may be awkward and may use approaches that require specialized knowledge of decision trees or languages. Furthermore, such approaches may not support incremental changes, further adding to the burden of maintaining the ACLs.

[0004] Therefore, what is needed is a system and method that addresses the above-identified issues.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Fig. 1 is a block diagram of an access control list.

[0006] Fig. 2 is a flow chart illustrating one embodiment of a method for inserting an access control rule into an access control list.

[0007] Fig. 3 is a flow chart illustrating one embodiment of a method for merging independent rule blocks.

[0008] Fig. 4 is a flow chart illustrating one embodiment of a method for determining a position for an access rule in an independent rule block.

[0009] Fig. 5 is a diagram of an exemplary computing environment in which an access control list may be implemented.

DETAILED DESCRIPTION

[0010] The present disclosure relates generally to networked computing environments and, more specifically, to a method and system for automatically creating and updating access control rule lists. It is understood, however, that the following disclosure provides many different embodiments or examples. Specific examples of components and arrangements are described below to simplify the present disclosure. These are, of course, merely examples and are not intended to be limiting. In addition, the present disclosure may repeat reference numerals and/or letters in the various examples. This repetition is for the purpose of simplicity and clarity and does not in itself dictate a relationship between the various embodiments and/or configurations discussed.

[0011] Referring to Fig. 1, a block diagram of an access control list (ACL) 100 is illustrated. The ACL 100 may include one or more independent rule blocks (IRBs) 110, 120, 130. The IRBs 110, 120, 130 may each include an ordered group of access rules (e.g., Rule 1 and Rule 2 of 110). Here D1, D2, ..., D_m correspond to the domain of traffic attributes. Traffic attributes may be a source user of an access request, a source or destination address of a machine, or an associated protocol, port, or time of a request,

for example. A_r defines an action domain of the rules, which may be either "allow," "deny," or other actions. The ACL 100 may be accessed in sequential order for each access request to find the first mapped rule. The action defined in the first mapped rule may be applied for that request. The order of the rules in the IRBs may be created and maintained by a creation/insertion method as will be described in greater detail below.

[0012] The access rules, Rule 1, Rule 2, ..., Rule i, within each block 110, 120, 130 may include an expression of one or more access control policies. The access control policies may be access policies for various network resources, for example. The access control rules Rule 1, Rule 2, ..., Rule i, may be applied to incoming and/or outgoing access requests. Each access control rule, Rule 1, Rule 2, ..., Rule i, may be defined in terms of attributes, Attr1, Attr2, ..., Attrm, which may fall under attribute domains D1, D2, ..., Dm, and an associated action, e.g., Action 1, for each rule, which may fall under the action domain Ar. All rules in an ACL may have the same attribute prototype.

[0013] An access request may include various attributes, which may fall under attribute domains, D1, D2, ..., Dm, and may match the attributes of a given rule. An action (e.g., Action1) may be applied for an access request if all the attributes of the request satisfy the rule-attributes Attr1, Attr2, Attrm, for a given rule. For example, Rule 1 of IRB 1 may express a policy that hypertext transfer protocol (http) sites may not be viewed between 9 AM and 5 PM. The rule corresponding to this policy may be expressed in terms of its attributes, {service, time} as, R1: *{http,9AM-5PM}* -> deny. This rule would not be applicable to an access request to view an http site at 7:00 PM, for example. In such case, the remainder of the ACL 100 may be examined in search of an applicable rule. However, an access request to view an http site at 12:30 PM would match the rule and may be denied without further processing of the ACL 100.

[0014] In some instances, more than one rule may be used to adequately express a given access policy. For example, a policy such as, "all http requests from employee X to any destination except www.xyz.com between 8:00 PM and 12:00 AM are allowed," may employ a first rule, R1: *{X, http, www.xyz.com, 8PM-12AM}* -> deny, followed by a second rule, R2: *{X, http, any, 8AM-12AM}* -> allow. Here Rule 1 and Rule 2 may need

to be processed in order for the original policy to be properly enforced. Thus, any new rules that may be inserted, or rules that may be deleted, may create a need to monitor or modify the ACL 100 in order to ensure that existing policy is not disturbed or contradicted. Additionally, without monitoring, rules may be inserted that are redundant.

[0015] In the present embodiment, to aid in maintaining the ACL 100, relationships may exist between rules that are helpful to recognize. Two rules may be said to be different rules if they are defined in different contexts. The context, or context space, of a rule refers the set of domain of attributes that a rule expresses. For example, the rule, R1: *({X,Y},{ftp,http},{9AM-5PM})->* allow is defined in the context space of *user* $\in$ *{X,Y} and service* $\in$ *{ftp, http} and time* $\in$ *{9AM-5PM}*. The rule, R2 = *({X,Z},{ftp,http},{3PM-9PM})->* allow, having a context space of *user* $\in$ *{X,Y} and service = {ftp, http} and time* $\in$ *{3PM-9PM}*, has a different context than R1 since the *user* domains and *time* domains are different.

[0016] Two contexts are disjoint if at least one of their attribute domains is mutually disjoint (e.g., different). For example, context 1: *{user = X,Y} and {service = ftp, http} and {time = 9AM-5PM}* and context 2: *{user = Z} and {service = ftp, http} and {time = 9AM-5PM}* are disjoint since the *user* domains are different. One context may overlap another. For example, context 1: *{user = X,Y} and {service = ftp} and {time = 9AM-5PM}* and context 2: *{user = Y} and {service = ftp, http} and {time = 3PM-8PM}* overlap since there is a common context, *{user=Y; service = ftp; time = 3PM-5PM}*, for all attributes. One context may also cover another. For example, context 1: *{user = X,Y} and {service = ftp, http} and {time = 9AM-5PM}* covers context 2: *{user = Y} and {service = http} and {time = 3PM-5PM}* since the domain of all attributes in context 2 are also defined in context 1. It may be said of this latter example that context 2 is a subset of context 1.

[0017] As previously stated, the ACL 100 of Fig. 1 may include IRBs 110, 120, 130. Each IRB may have the property that each access rule within an IRB containing two or more access rules overlaps with at least one other rule in the same IRB. The IRBs may also

have the property that each access rule within a given IRB is disjoint with each access rule of any other IRB.

[0018] Referring to Fig. 2, a flow chart illustrating one embodiment of a method 200 for inserting an access control rule into an access control list is illustrated. The method 200 may used as an iterative ACL creation and insertion tool to maintain the ACL 100 in a consistent state (e.g., having no redundancies or conflicts with policy).

[0019] In step 220, the list 100 may be determined to be empty, or may be determined to already contain one or more access control rules. The rules may be disposed in one or more independent rule blocks. If the list is determined to be empty in step 220, a new IRB may be created in step 230. The new rule may be inserted in step 240 and the IRB may contain only the new rule being inserted. If the list is not determined to be empty in step 220, the list may contain at least one IRB already. In this case, all the IRBs in the list which map to the new rule may be located. The new rule may be said to map to each IRB in the list that contains at least one rule with which the new rule is not disjoint (e.g. the two rules have an overlapped context as previously described).

[0020] The new rule may map to only a single IRB, or it may map to multiple IRBs. In the case of the new rule mapping to multiple IRBs, the IRBs may need to be merged into a single new IRB in step 270. An exemplary method for merging the mapped IRBs will be described in greater detail below with respect to Fig. 3. The mapped IRBs may be merged in such a way that redundancies and contradictions are not created, or are removed, as will be described below. Following the merging of the mapped IRBs in step 270, or concurrently with the merging, the new rule is inserted into the new IRB in step 280. The position of the new rule within the new rule block may be based on its position within the merged blocks as described in greater detail below.

[0021] Referring to Fig. 3, a flow chart of one embodiment of a method 300 for merging independent rule blocks and inserting a new rule is illustrated. In some embodiments, the method 300 may incorporate both the merging step 270 and insertion step 280 of Fig. 2. The method 300 may also function as an insertion method to form a new independent rule block. For example, if the set of mapped IRBs is determined to be

empty in step 320 (e.g., there may be no mapped IRBs), a new IRB formed in step 330 may include a header, the new rule, and a trailer. However, in the case of an empty mapped set of IRBs in step 320, the header and trailer may contain nothing and the new IRB may include only the new rule in step 330.

[0022] If the mapped IRB set is determined to be not empty in step 320, starting with the first IRB in step 350, which may be any IRB in the set, a position within the IRB for the new rule may be found in step 355. The position within the IRB for the new rule may be a position that does not introduce redundancies or contradictions. This may involve resolving conflicts, or removing redundant and/or conflicting rules from the IRB before a position for the new rule may be determined. One method for determining a position for a new rule within an IRB will be described in greater detail below with respect to Fig. 4. In some instances, there may be no position for a new rule within an IRB. For example, the rule may not map to an IRB or it may map to an IRB but be redundant. If this is determined in step 360, yet the IRB is be needed to enforce a desired access policy, the IRB may be removed from the mapped set in step 365, and hence may not be merged but will remain within the ACL 100.

[0023] In a case where a position can be found for the new rule as determined in step 360, the IRB may be divided into a header and a trailer. The header may be the ordered list of all the rules in the IRB which have a position ahead of the new rule as determined in step 355. The trailer may be the ordered list of rules which have a position behind the new rule as determined in step 355. In step 380, the header from the current IRB may be placed following the header or headers from any previous IRB evaluations. In step 385, the trailer from the current IRB may be placed following the trailer or trailers from any previous IRB evaluations.

[0024] The process of evaluating IRBs in step 355 (to find a position for the new rule) and either splitting the IRB and adding it to the header in step 380 and/or trailer in step 385, or discarding it from the mapped set in step 365 may be repeated until there are no IRBs remaining in the set as determined in step 370. The new IRB, including the new rule, may then be formed in step 330 by creating a new IRB from the header, followed

by the new rule, followed by the trailer. As stated previously, it is possible for the header, trailer, or both to be empty sets.

[0025] Referring to Fig. 4, a flow chart of one embodiment of a method 400 for determining a position for an access rule in an IRB is illustrated. A counter variable may be used (e.g., i) beginning in step 410 to traverse the ordered list of rules within an IRB. If i represents the end of the IRB as determined in step 415, the may be no position for the new rule in step 420. If i does not represent the end of the IRB as determined in step 415, the current rule (the ith rule in the IRB) may be tested for a disjoint context with the new rule in step 430. If the two rules are disjoint, the counter i may be incremented in step 435, and the process may return to 415. If the two rules are determined to be not disjoint in step 430, a determination may be made as to whether the new rule is a subset of the ith rule in step 440. If so, a determination may be made as to whether the two rules perform the same action in step 445. If so, there may be no position in the IRB for the new rule in step 450. If the two rules do not perform the same action as determined in step 445, there may be a conflict which may be resolved in step 452. If the new rule takes priority as determined in step 455, i may be the position of the new rule in step 460. In other words, the new rule takes the ith position and the ith rule moves to the (i+1)th position. If the ith rule takes priority in step 445, there may be no position in the IRB for the new rule.

[0026] Returning to step 440, if the new rule is not a subset of the ith rule, but instead the ith rule is a subset of the new rule as determined in step 465, and the two rules perform the same action as determined in step 470, the ith rule may removed in step 477 and i may be the position of the new rule in step 460. If the two rules do not perform the same action as determined in step 470, there may be a conflict that may be resolved in step 472. If the new rule takes priority as determined in step 475, the ith rule may be removed and i may be the position of the new rule. In other words, the new rule takes the ith position and the ith rules moves to the (i+1)th position. If the ith rule takes priority as determined in step 475, the position of the new rule may be set to i+1.

[0027] In the case where neither rule is a subset of the other as determined in steps 440 and 465 and the two rules do not perform the same action as determined in step 482, a conflict may be resolved in step 485. If the new rule takes priority as determined in step 490, i may become the position for the new rule. If the two rules perform the same action as determined in step 482 or if the ith rule takes priority as determined in step 490, i may be increment in step 491 and the process may repeat.

[0028] The process of resolving conflicts, or determining a priority, in steps 452, 472, and 485 may be based on user input as the ACL 100 is being created or updated. The user may have the option of deciding during modification of the ACL 100 which of two conflicting rules to keep and enforce. The priority decisions may also be based on predefined rules. For example, the newer rule may always take priority, reflecting a presumption that the newer rule reflects a newer or changed access policy. It is also possible for rules to be defined for some conflicts and to request user input or selection for other conflicts.

[0029] The process of removing rules (e.g., step 477) may not require any user intervention or reordering of the remaining access rules. The removal of a rule as in step 477, or the removal of a rule from the ACL 100 in other situations, may be done by locating the rule and removing it from its IRB. The location of a rule may be determined based on a unique identifier or ID number for each rule, for example, or a location may be based on other querying or location techniques.

[0030] The insertion method 300, and position determining method 400 may be flexible in that any number of attributes may be added to update an ACL policy. IRBs may enable rule blocking, which may speed up finding a matching rule. Well-known techniques of prioritization may be utilized for even faster processesing. Because each IRB may be independent, the IRBs may be placed in any position within the ACL 100. However, the IRBs may also be arranged based on frequency of access to reduce average search times and increase overall performance of the ACL 100.

[0031] The examples illustrated show access rules based on binary outcomes such as "allow," or "deny." However the ACL 100 may also be configured to allow a greater

variety of outcomes. For example, a "redirect request" rule or a "log access attempt" rule may also be utilized in the ACL 100 to extend the capabilities of the ACL 100.

[0032] Referring to Fig. 5, a diagram of an exemplary environment 500, in which an access control list may be used. A user may enter an access control policy using a policy specification language 510 and a translator 520 may convert the policy into one or more access control rules 530, which may be passed to a rule insertion engine 540. The rule insertion engine may update or create an ACL (e.g., the ACL 100 of Fig. 1), which may be stored in a rule base 550. A rule enforcing engine 560 may monitor network traffic and allow or deny access requests from network 570 to network 580 based on the result of accessing the rule base 550, for example. The user may delete an access policy by entering a policy id 590. The rule insertion engine 540 may remove the corresponding access control rules from the rule base 550. Although one-way communication is shown between the various components of environment 500, it is understood that two-way communication may also be possible along each communication path.

[0033] The policy specification language 510 and translator 520 may be used to enter access policies which result in a rule-based specification. The policy specification language 510 may be a meta-language, for example. The translator 520 may be a compiler for policy statements defined in a meta-language. A GUI (not shown) may also be used, which accepts attributes and their ranges in addition to the decisions supported for the ACL 100. Other methods or tools may be used to generate the rules 530, or they may be generated manually. The policy specification language 510, translator 520, and rule insertion engine 540 may be software based and executable on a processor, which may be part of a larger computer system. A computer system may include, but is not limited to, a personal computer (PC), a server, a terminal, a personal digital assistant (PDA), or a mobile phone. The computer system may also make use of volatile memory such as RAM, for example, and non-volatile memory such as a hard disk drive or a flash memory device, for example.

[0034] The access rules 530 may be access control rules as previously described. The rule insertion engine 540 may create or update ACLs based on the methods previously

described, which may prevent redundancies, conflicts, or other disorders of the access rules. The rules in the ACL 100 may be stored in a rule base 550. The rule insertion engine 540 and the rule base 550 may be hardware or software based and may be executable on a processor or computer system as previously herein described. The rule base 550 may also be a remotely stored database accessible via a network, such as a local area network (LAN), a wide area network (WAN), a wireless network, or the Internet, for example.

[0035] The rule enforcing engine 560 may be software or hardware based and may reside on the same system, or a different system, than the rule base 550. The rule enforcing engine may also reside on a router or network access point, for example. The rule enforcing engine 560 may check access requests, such as an access request from network 570 to network 580. An access request may be checked against the ACL 100 stored in the rule base 550 to determine whether to allow or deny the request. The networks 570, 580 may be single computers, or processors, a single networked resource, or an entire network. Similarly, the rule enforcing engine 560 may interact with more networks or devices than shown.

[0036] An access policy or a single access control rule may be removed from the rule base 550 by the rule insertion engine 540 as previously described. Policy deletion 590 may be a part of the same or different componentry as previously described with respect to rule insertion. Policy deletion may be hardware or software based, for example, and may be executable on a processor or computer system.

[0037] Those skilled in the art should appreciate that they may readily use the present disclosure as a basis for designing or modifying other processes and structures for carrying out the same purposes and/or achieving the same advantages of the embodiments introduced herein. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the present disclosure, and that they may make various changes, substitutions and alterations herein without departing from the spirit and scope of the present disclosure.